Python Web frameworks

SBNA-Report

# Project Overview:

To describe about the Python Web frameworks

# Objectives

* To understand and analyze the Python web frameworks
* To understand the significance of each frameworks
* List learning and content standards here

# Requirements/Task(s):

Basics of python and web frameworks

# Research Notes:

There are several Web frameworks related to python Like Tornado, Django, flask and many full stack web frameworks and server frameworks.

But in terms of popularity among the community Django proves to be the best one as it is fast and easy to develop.

But many web developers feel that Django is not the future as there is a huge competition from frameworks like Nodejs and Denojs as they have more code readability and developers ease along with performance

But the framework has a lot of potential for the ones who love python

**Django:**

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django, like other modern web frameworks, supports model-view-controller (MVC) design rule. The MVC programming paradigm allows programmers to keep a web application’s user interface (UI) and business logic layers separated. The approach further helps programmers to simplify and speed up development of large web applications by separating their user interface and business logic layers. Django further allows programmers to reuse the same business logic across multiple projects.

As an open source web framework for Python, Django helps developers to reduce web application development cost significantly. But it is supported by a large and active community of developers. The members of the Django community update new plug-ins and code snippets regularly to simplify web application development. The developers can easily speed up custom web application development by taking advantage of these resources uploaded by members of the Django community. The members of the community even help developers to resolve common web application development issues and problems.

The robust features provided by Django make it easier for programmers to build large and complex web applications. But the developers do not need some of these advanced features while developing small and simple web applications. The additional features even impact the performance of small web applications adversely. Hence, Python programmers have to explore ways to improve the performance of small web application while using Django.

Most web frameworks improve the performance of web applications by making each process handle multiple requests at a time. But Django, unlike other modern web framework, does not enable individual processes to handle multiple requests simultaneously. Hence, programmers have to explore ways to make individual processes handle multiple requests efficiently and quickly at a time.

he ORM system provided by Django makes it easier for developers to work with multiple databases and perform common database operations. But the ORM system used by the web framework lacks some of the robust features provided by other widely used ORM systems. Also, it does not enable developers to take advantage of the robust the Python SQL toolkit – SQLAlchemy – fully.

**Flask:**

Flask is considered more Pythonic than the Django web framework because in common situations the equivalent Flask web application is more explicit. Flask is also easy to get started with as a beginner because there is little boilerplate code for getting a simple app up and running.

* Flask has a lightweight and modular design, so it easy to transform it to the web framework you need with a few extensions without weighing it down
* ORM-agnostic: you can plug in your favorite ORM e.g. SQLAlchemy.
* Basic foundation API is nicely shaped and coherent.
* Flask documentation is comprehensive, full of examples and well structured. You can even try out some sample application to really get a feel of Flask.
* It is super easy to deploy Flask in production (Flask is 100% WSGI 1.0 compliant”)
* HTTP request handling functionality
* High Flexibility
* The configuration is even more flexible than that of Django, giving you plenty of solution for every production need

**Pyramid:**

Pyramid is a general, open source, web application development framework built in python. It allows python developer to create web applications with ease.

Pyramid is backed by the enterprise knowledge Management System KARL (a George Soros project).

Pyramid starts with a minimal installation that can be extended when needed. It’s worth noting that it’s a part of the Pylons project, which integrates web-related technologies. These are some of Pyramid’s most important characteristics:

* offers synchronous request processing,
* views contexts can be defined with functions as well as classes,
* there is no specific ORM but SQLAlchemy is recommended,
* doesn’t force any coding style or project architecture – TIMTOWTDI,
* offers good documentation with tutorials and examples,
* no specific HTML template engine is delivered but Chameleon is recommended,
* an interesting custom routing system allows multiple views to match one URL,
* it’s WSGI compliant,
* extensive static files support – files serving, URL routing for static files,
* can be extended with external modules, e.g. Cornice for REST APIs, aiopyramid with async support.

This framework might be a good choice in cases when you don’t want to spend time learning custom frameworks solutions (like ORM) but still need an extensive tool for building software, as Pyramid supports usage of many well-known stand-alone solutions. It has good scaling capabilities – advertises itself as a framework that can “start small and finish big”.

**Web2py:**

Web2py focuses on security, development speed and ease of use. It offers a lot of features out of the box: a web server, database, admin panel, wiki or grid widgets. Framework main characteristics are:

* synchronous request processing,
* custom Database Abstraction Layer (DAL) that acts as ORM,
* forces an MVC structure,
* functions and classes can be used for creating Controllers,
* strict “There should be only one way of doing things” philosophy,
* rich documentation with a lot of examples,
* custom HTML engine that allows Python code to be used in the templates,
* custom routing - URL function that generates internal paths for the actions and static files,
* support for the WSGI standard, but it’s possible to use CGI (Common Gateway Interface), FastCGI, GAE (Google App Engine) or other,
* offers static file routing and streaming during development,
* has built-in REST services but requires the Tornado framework for Web Socket usage.

Web2py was highly inspired by Ruby on Rails and Django frameworks and takes what’s the best from both of them. It can be a good choice for programmers who want to migrate from Ruby or for ones that are bored with Django but are looking for another big and feature-rich framework. It offers an “admin” app, which acts as a web-based IDE for application development and management (e.g. app creations, code editor). It is also supported by PyCharm. In general, Web2Py doesn’t lack any functionalities that Django has. These two frameworks can be used to fulfil the same tasks. Web2Py is younger, and it has a smaller community than Django, so it might be a little harder to find help in case of trouble.

**Tornado:**

Tornado is a Python web framework and asynchronous networking library, originally developed at FriendFeed (a social aggregation site). Thanks to that, it offers a built-in integration with social services such as Google, Facebook, and Twitter. The integration with other frameworks and libraries is also possible: Twisted, asyncio or even WSGI applications. Tornado’s features:

* offers a lot of generic classes that can be used for creating the application, e.g. Router, or SocketHandler for websockets,
* custom HTML template engine,
* clear and easy-to-read documentation,
* functions and classes can be used for defining actions and handling requests,
* custom routing handling – offers generic classes than can be used for route creation,
* it supports WSGI, but it’s not recommended – the user should use Tornado’s own interfaces instead,
* out-of-the-box WebSockets support, authentication (e.g. via Google), and security features (like cookie signing or XSRF protection),
* no additional tools are needed for REST API creation.

The framework should work well in cases where there are a lot of incoming connections that can be handled quickly or in real-time solutions, e.g. chats. Tornado tries to solve the c10k problem so high processing speed is a priority. Another advantage of Tornado is its native support for social services. This framework won’t be a good choice for creating standard CRUD sites or big business applications, as it wasn’t designed to be used that way. For bigger projects, it can be integrated with WSGI applications as a part of their bigger structure and take care of tasks that require high handling speeds.

# Summarize what you learned:

There are many frameworks from above all the Tornado, Django and Flask seems to be appealing and easy to develop. But in terms of popularity and community support Django seems to be a better option